Linear Regression and Regularization

Applied Machine Learning
Derek Hoiem

Dall-E

Today's Lecture

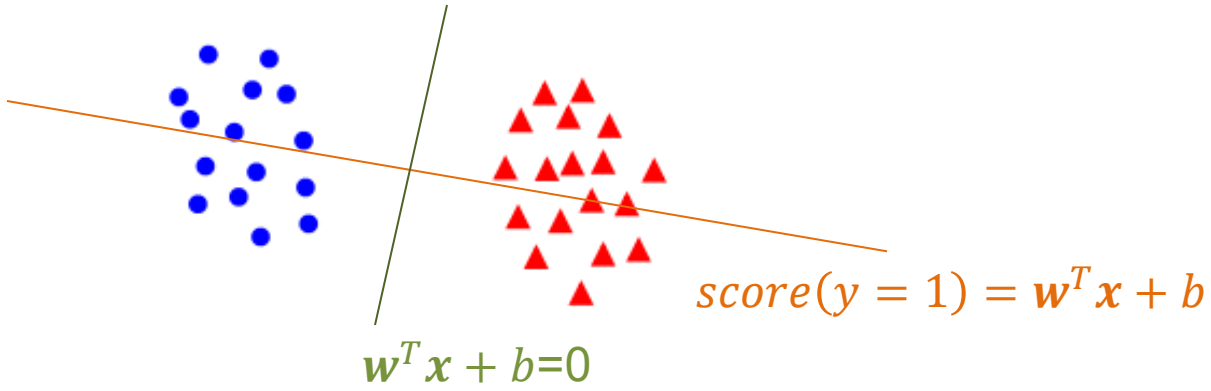
- Linear regression
- Regularization
  - What is it
  - How it affects models
  - How to select regularization parameters

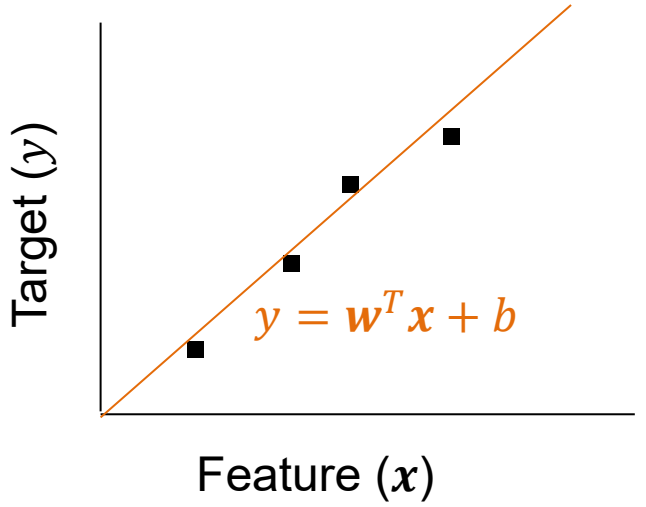
Linear Models

- A model is **linear** in x if it is based on a weighted sum of the values of x (optionally, plus a constant)

$$\boldsymbol{w}^T\boldsymbol{x} + b = \left[\sum_i w_i x_i\right] + b$$

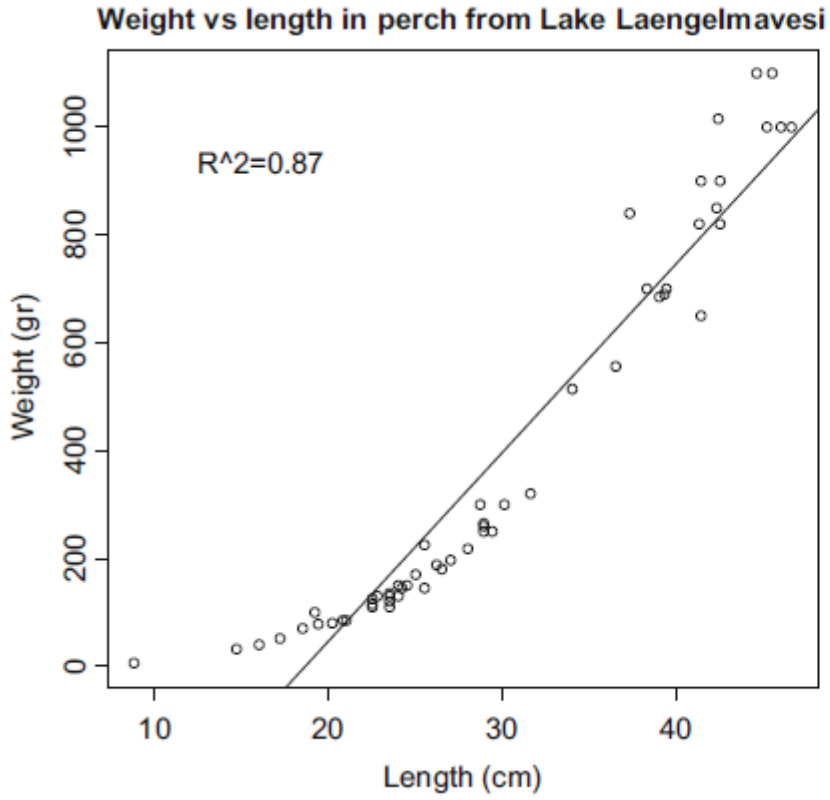
- A **linear classifier** projects the features onto a score that indicates whether the label is positive or negative (i.e., one class or the other). We often show the boundary where that score is equal to zero.
- A **linear regressor** finds a linear model that approximates the prediction value for each set of features.

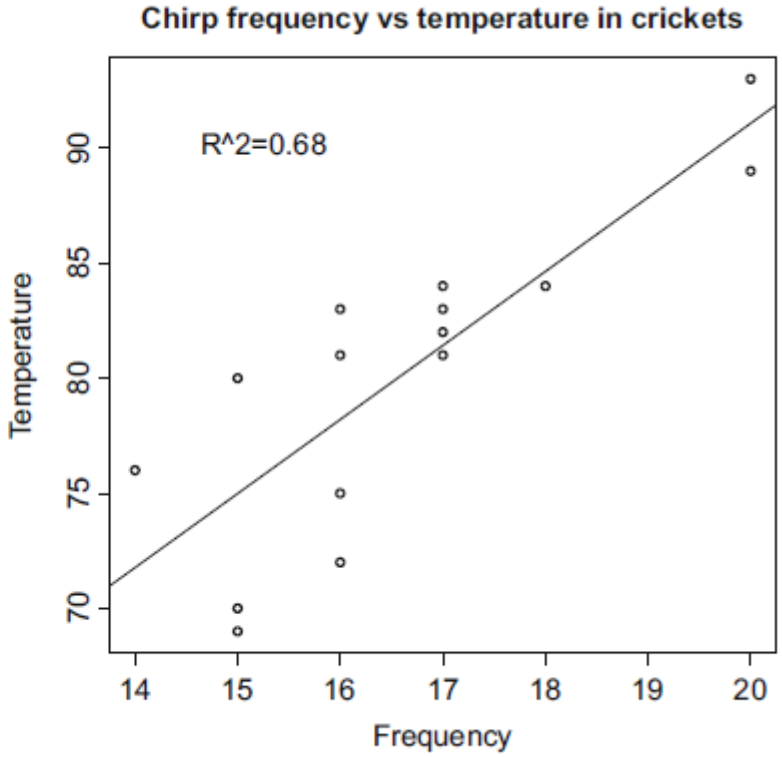




Linear regression

- Fit linear coefficients to features to predict a continuous variable





R^2: $1 - \sum_i (f(X_i) - y_i)^2 \,/\, \sum_i (y_i - \bar{\boldsymbol{y}})^2$

- R^2 close to zero indicates very weak relationship
- R^2 close to 1 indicates y very linearly predictable from **x**

Linear Regression algorithm

- Training: find $\boldsymbol{w}$ that minimizes sum square difference between target and prediction

$$w^* = \underset{w}{\operatorname{argmin}} \sum_n (\boldsymbol{w}^T \boldsymbol{x}_n - y_n)^2 + r(\boldsymbol{w})$$

Loss due to missing the target

Loss due to parameter complexity

- Prediction

$$y = \boldsymbol{w}^T \boldsymbol{x}$$

Linear regression

Model/Prediction

$$Ax + b = y \rightarrow A[x; 1] = y$$

If y is 1 dim: $a^T[x; 1] = y$

Solve with least squares:

Training (least squares)

$$a^* = \underset{a}{\operatorname{argmin}} \sum_n^N (a_0 x_{n0} + \ldots + a_m x_{nm} - y_n)^2$$

$$= \underset{a}{\operatorname{argmin}} (Xa - y)^T (Xa - y)$$

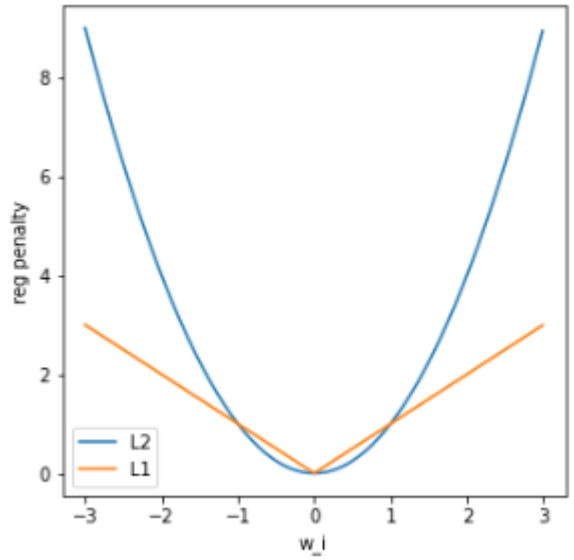
X: $n \times m$, a: $m \times 1$, y: $n \times 1$

$$\Rightarrow a^* = X^\dagger y \qquad X^\dagger = (X^T X)^{-1} X^T y$$

Training Linear Regression

$$w^* = \underset{w}{\operatorname{argmin}} \sum_n (w^T x - y_n)^2 + r(w)$$

- L2 regularization: $r(w) = \lambda \|w\|_2^2 = \lambda \sum_i w_i^2$
- L1 regularization: $r(w) = \lambda \|w\|_1 = \lambda \sum_i |w_i|$



L2 strongly penalizes big weights

L1 penalizes increasing the magnitude of big and small weights the same

L1 regularization can be used to select features

L2 linear regression is least squares and not hard to implement, but you can use a library.

How to select hyperparameters

"Hyperparameters" are part of the objective function, not something that training data can fit.

E.g., the regularization weight λ is a hyperparameter

How to select hyperparameters

1. For λ in {1/8, ¼, ½, 1, 2, 4, 8}:
   a. Train model with λ using training set
   b. Measure and record performance on validation set
2. Choose λ with best validation performance
3. (Optional) re-train on train + val sets
4. Test final model on the test set

Tips

- In many cases, you want to vary parameters by factors, e.g. times 2 or times 5 for each candidate
- You can start search broad and then narrow, e.g. if ¼ and ½ are the best two, then try 3/8

Linear Regression Walkthrough

Diabetes Dataset from sklearn

Pause (2 min) and think/stretch

- Suppose I have two features, x1 and x2, that are each predictive of y.
  - $x2 = x1 + \epsilon$, where epsilon is a small amount of Gaussian noise

  Suppose we train a linear regressor, giving weights for each feature.

  Q: Will the weights of x1 and x2 be similar:

  a. under L2-regularized regression?
  b. under L1-regularized regression?

$$w^* = \underset{w}{\operatorname{argmin}} \sum_n (w^T x_n - y_n)^2 + r(w)$$

L2 regularization: $r(w) = \lambda \|w\|_2^2 = \lambda \sum_i w_i^2$
L1 regularization: $r(w) = \lambda \|w\|_1 = \lambda \sum_i |w_i|$

Hyper-parameter selection with multiple variables

- Basic methods
  - Grid search: try all possible combinations
    - Becomes infeasible for more than two parameters
  - Line search: optimize each parameter by itself sequentially
    - Does not account for dependencies between parameters
  - Randomized search: randomly generate parameters within a range
    - Often best strategy for 3+ parameters

[Article by Prof. Arindam Banerjee]

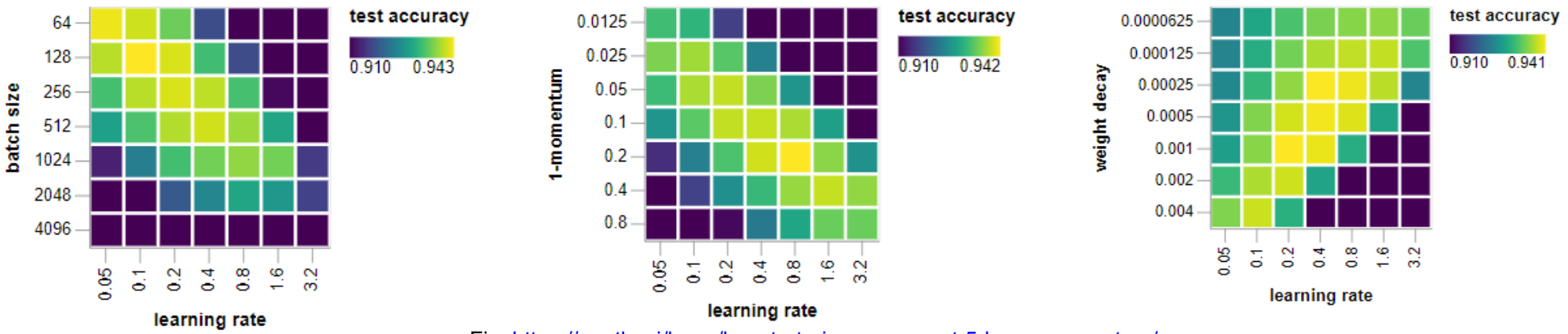


Fig: https://myrtle.ai/learn/how-to-train-your-resnet-5-hyperparameters/

Other forms of hyper-parameter selection

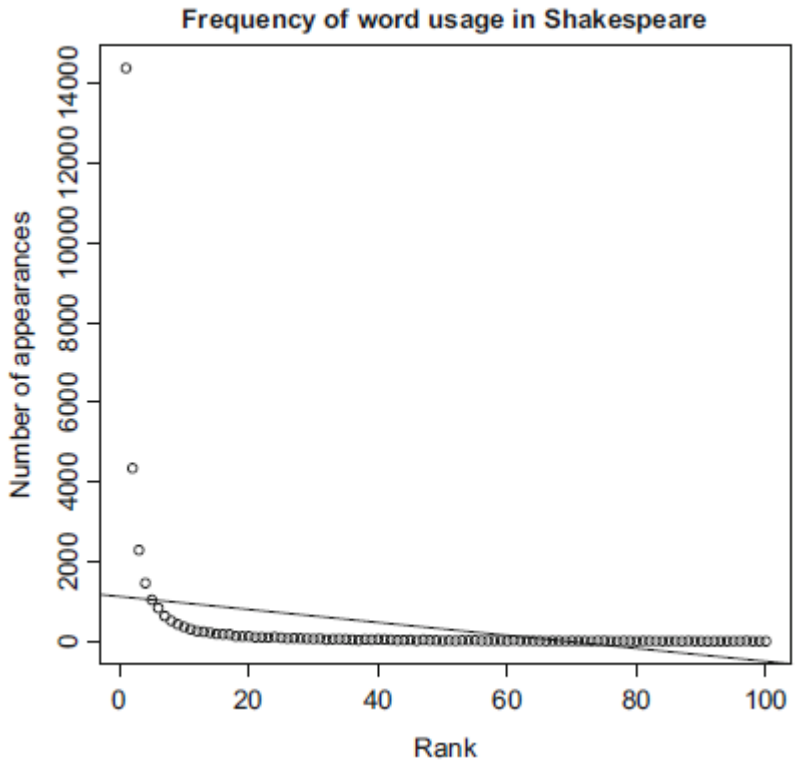
- Cross-validation
  1. Split training set into 5 parts (for 5-fold cross-validation)
  2. Train on 4/5 and train on remaining fifth
  3. Repeat five times, using a different fifth for validation each time
  4. Choose hyperparameters based on average validation performance

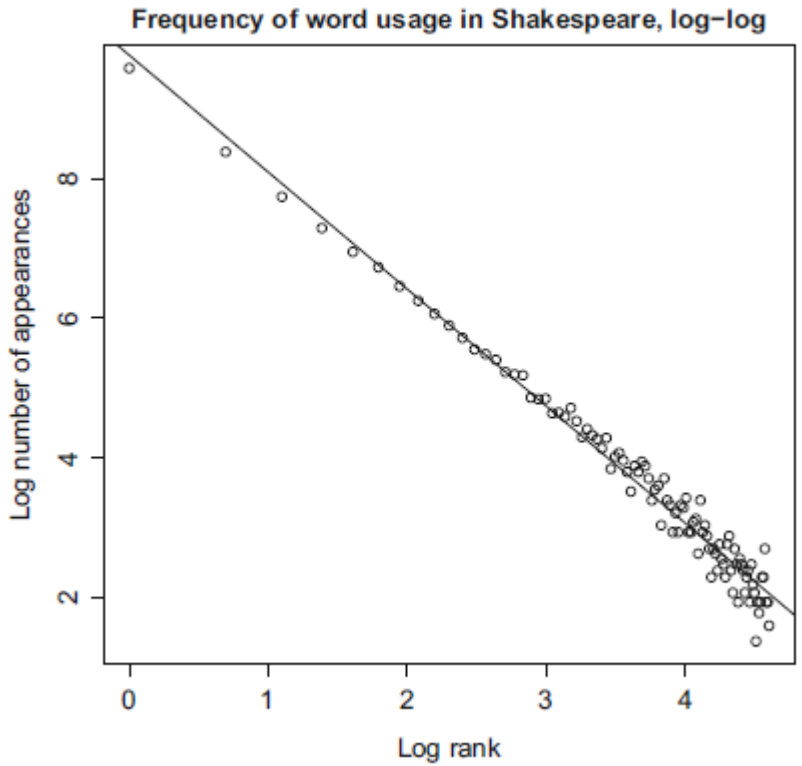
  – Useful when you have limited training data

- Leave-one-out cross-validation (LOOCV) is an extreme where for N data points, you have N splits (one held out for validation each time)

Transforming variables

- Sometimes you need to transform variables before fitting a linear model
  - Helpful to plot histograms or distributions of individual features to figure out what transformations might apply

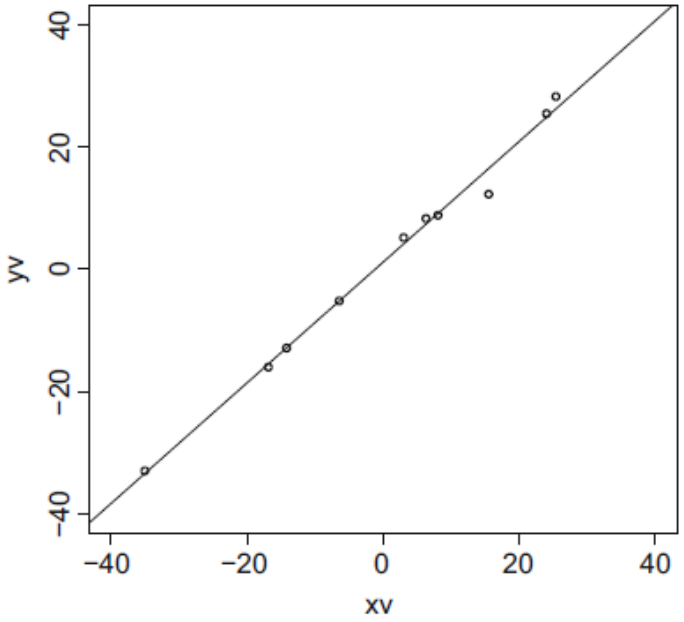


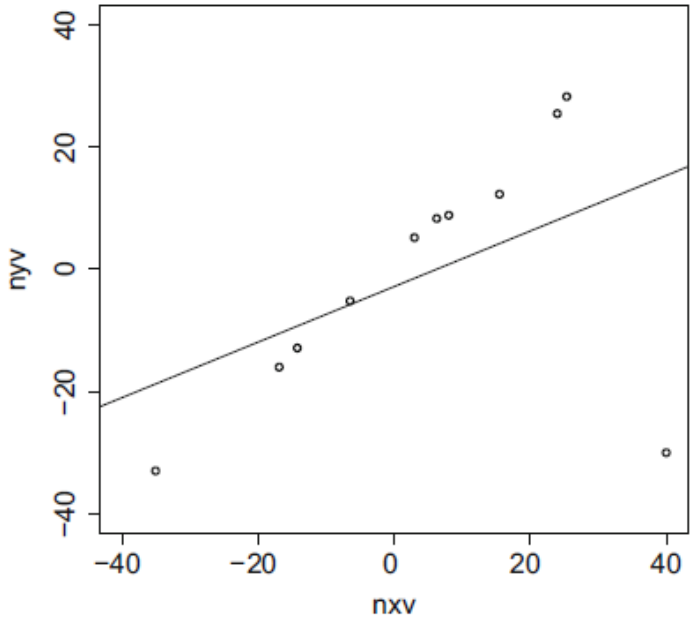


Example from Forsyth AML

Linear regression can be sensitive to outliers

- There are robust forms that estimate a weight on each sample, e.g. m-estimation
- Minimizing the sum of absolute error does not have this problem, but is a harder optimization

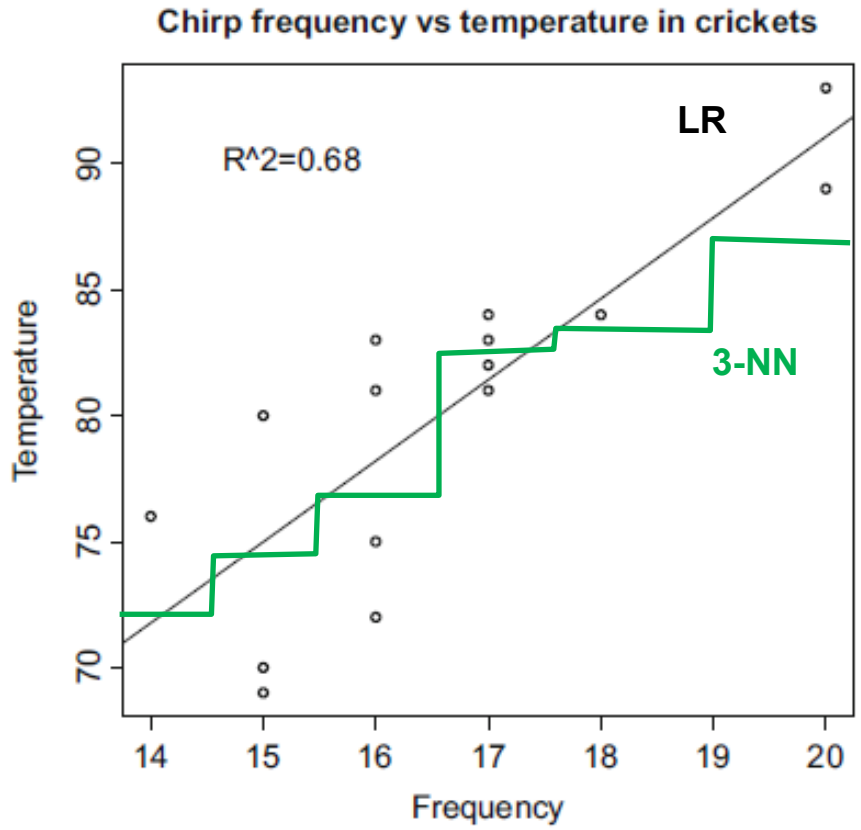




Example from Forsyth AML

Linear regression vs KNN

- KNN can fit non-linear functions
- Only linear regression can extrapolate
- Linear regression is more useful to explain a relationship



Linear Regression Summary

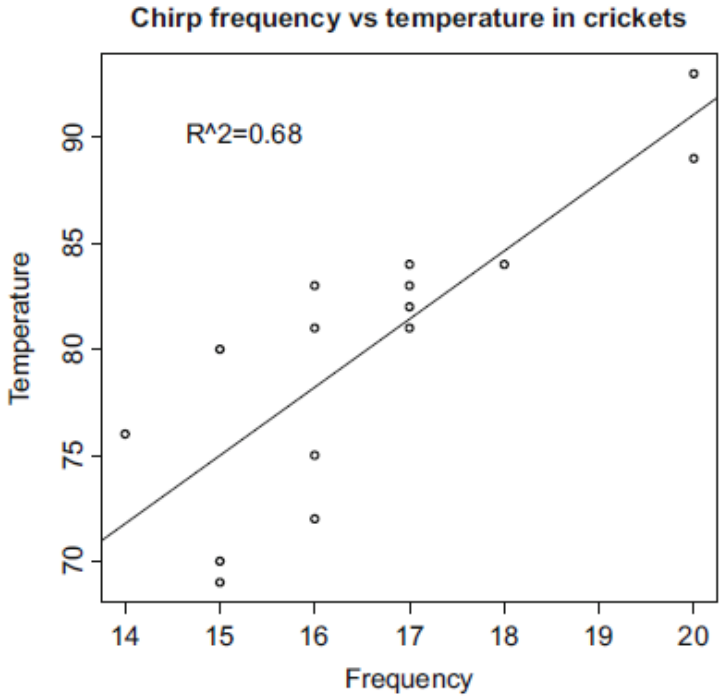
Linear regression is used to explain data or predict continuous variables in a wide range of applications

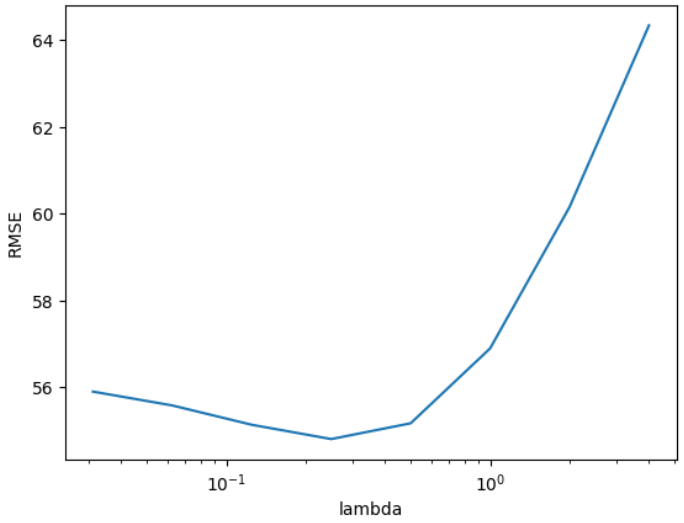
- Key Assumptions
  - y can be predicted by a linear combination of features
- Model Parameters
  - One coefficient per feature (plus one for y-intercept)
- Designs
  - L1 or L2 or elastic (both L1 and L2) regularization weight
  - Different objective functions (e.g. squared error, absolute error, m-estimation)
- When to Use
  - Want to extrapolate
  - Want to visualize or quantify correlations/relationships
  - Have many features
- When Not to Use
  - Relationships are very non-linear (requires transformation or feature learning first)

Things to remember

- Linear regression fits a linear model to a set of feature points to predict a continuous value
  - Explain relationships
  - Predict values
  - Extrapolate observations

- Regularization prevents overfitting by restricting the magnitude of feature weights
  - L1: prefers to assign a lot of weight to the most useful features
  - L2: prefers to assign smaller weight to everything





Next week

- Linear classifiers
- Probability and Naïve Bayes Classifier